EXHIBIT 1

```
                                                    Date
/*
 * @(#)MktInfoPane.java     1.0    Masked.
 *
 * C(  Date                  WIT Capital, Inc. All Rights Reserved.
 *     Masked.
 * This software is the confidential and proprietary information of WIT
 * Capital, Inc. ("Confidential Information"). You shall not
 * disclose such Confidential Information and shall use it only in
 * accordance with the terms of the license agreement you entered into
 * with Sun.
 *
 * WIT MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE
SUITABILITY OF THE
 * SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
TO THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR
 * PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR ANY
DAMAGES
 * SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
DISTRIBUTING
 * THIS SOFTWARE OR ITS DERIVATIVES.
 *
 */

package com.witcapital.dsm.client;

import com.sun.java.swing.*;
import com.sun.java.swing.JButton;
import com.sun.java.swing.JComboBox;
import com.sun.java.swing.JLabel;
import com.sun.java.swing.JPanel;
import com.sun.java.swing.border.MatteBorder;
import com.sun.java.swing.event.EventListenerList;

import java.awt.*;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Enumeration;
import java.util.Observable;
import java.util.Observer;
```

```
import java.util.Properties;
import java.util.Vector;

import com.witcapital.fwk.*;
import com.witcapital.biz.*;

/*
 * The view of an issue's market info.
 * 1.0 98/06/18
 * @author Christopher Buist
 */
public class MktInfoPane extends com.sun.java.swing.JPanel
implements java.util.Observer
{
    /**
     * The collection of registered objects.
     */
    private Registry registry;

    /**
     * The collection of WITUsers.
     */
    private WITUsers  witUsers;

    /**
     * The collection of Preferences.
     */
    private Preferences  preferences;

    /**
     * The collection of Issues.
     */
    private Issues  issues;

    /**
     * The collection of MktInfos.
     */
    private MktInfos  mktInfos;

    /**
     * The model on display.
     */
    private MktInfo mktInfo;

    /**
     *
```

```
*/
private TitleBarPane paneTitleBar;

/**
 * Creates the view of an issue's market info.
 */
    public MktInfoPane( )
    {
            super();
            //{{INIT_CONTROLS
            setLayout(new BorderLayout(0,0));
            setSize(452,102);
            setFont(new Font("Dialog", Font.PLAIN, 11));
            setForeground(new Color(0));
            setBackground(new Color(-3355444));
            jPanel1 = new com.sun.java.swing.JPanel();
            jPanel1.setOpaque(false);
            GridBagLayout gridBagLayout;
            gridBagLayout = new GridBagLayout();
            jPanel1.setLayout(gridBagLayout);
            jPanel1.setBounds(0,25,452,77);
            jPanel1.setFont(new Font("Dialog", Font.PLAIN, 12));
            jPanel1.setForeground(new Color(0));
            jPanel1.setBackground(new Color(-3355444));
            add("Center", jPanel1);
            paneIssueInfo = new com.sun.java.swing.JPanel();
            paneIssueInfo.setOpaque(false);
            paneIssueInfo.setLayout(new GridLayout(1,4,0,0));
            paneIssueInfo.setBounds(0,0,20,40);
            paneIssueInfo.setFont(new Font("Dialog", Font.PLAIN, 12));
            paneIssueInfo.setForeground(new Color(0));
            paneIssueInfo.setBackground(new Color(-3355444));
            GridBagConstraints gbc;
            gbc = new GridBagConstraints();
            gbc.gridy = 0;
            gbc.gridwidth = 4;
            gbc.weightx = 1.0;
            gbc.fill = GridBagConstraints.HORIZONTAL;
            gbc.insets = new Insets(0,0,0,0);
            ((GridBagLayout)jPanel1.getLayout()).setConstraints(paneIssueInfo, gbc);
            jPanel1.add(paneIssueInfo);
            labelSymbol = new com.sun.java.swing.JLabel();
            labelSymbol.setText("symbol");
            labelSymbol.setBounds(0,0,113,25);
            labelSymbol.setFont(new Font("Dialog", Font.BOLD, 11));
            labelSymbol.setForeground(new Color(0));
```

```
labelSymbol.setBackground(new Color(-3355444));
paneIssueInfo.add(labelSymbol);
labelDescription = new com.sun.java.swing.JLabel();
labelDescription.setText("description");
labelDescription.setBounds(0,0,113,25);
labelDescription.setFont(new Font("Dialog", Font.BOLD, 11));
labelDescription.setForeground(new Color(0));
labelDescription.setBackground(new Color(-3355444));
paneIssueInfo.add(labelDescription);
jLabel2 = new com.sun.java.swing.JLabel();
jLabel2.setBounds(0,0,113,25);
jLabel2.setFont(new Font("Dialog", Font.BOLD, 11));
jLabel2.setForeground(new Color(0));
jLabel2.setBackground(new Color(-3355444));
paneIssueInfo.add(jLabel2);
jLabel3 = new com.sun.java.swing.JLabel();
jLabel3.setBounds(0,0,113,25);
jLabel3.setFont(new Font("Dialog", Font.BOLD, 11));
jLabel3.setForeground(new Color(0));
jLabel3.setBackground(new Color(-3355444));
paneIssueInfo.add(jLabel3);
labelOpen = new com.sun.java.swing.JLabel();
labelOpen.setText("Open");
labelOpen.setBounds(0,0,113,25);
labelOpen.setFont(new Font("Dialog", Font.PLAIN, 11));
labelOpen.setForeground(new Color(0));
labelOpen.setBackground(new Color(-3355444));
gbc = new GridBagConstraints();
gbc.gridy = 1;
gbc.weightx = 1.0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.insets = new Insets(0,0,0,0);
((GridBagLayout)jPanel1.getLayout()).setConstraints(labelOpen, gbc);
jPanel1.add(labelOpen);
labelHi = new com.sun.java.swing.JLabel();
labelHi.setText("Hi");
labelHi.setBounds(113,0,113,25);
labelHi.setFont(new Font("Dialog", Font.PLAIN, 11));
labelHi.setForeground(new Color(0));
labelHi.setBackground(new Color(-3355444));
gbc = new GridBagConstraints();
gbc.gridy = 1;
gbc.weightx = 1.0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.insets = new Insets(0,0,0,0);
((GridBagLayout)jPanel1.getLayout()).setConstraints(labelHi, gbc);
```

```
                jPanel1.add(labelHi);
                labelLastPrice = new com.sun.java.swing.JLabel();
                labelLastPrice.setText("Last");
                labelLastPrice.setBounds(226,0,113,25);
                labelLastPrice.setFont(new Font("Dialog", Font.PLAIN, 11));
                labelLastPrice.setForeground(new Color(0));
                labelLastPrice.setBackground(new Color(-3355444));
                gbc = new GridBagConstraints();
                gbc.gridy = 1;
                gbc.weightx = 1.0;
                gbc.fill = GridBagConstraints.HORIZONTAL;
                gbc.insets = new Insets(0,0,0,0);
                ((GridBagLayout)jPanel1.getLayout()).setConstraints(labelLastPrice,
gbc);
                jPanel1.add(labelLastPrice);
                labelChange = new com.sun.java.swing.JLabel();
                labelChange.setText("Change");
                labelChange.setBounds(339,0,113,25);
                labelChange.setFont(new Font("Dialog", Font.PLAIN, 11));
                labelChange.setForeground(new Color(0));
                labelChange.setBackground(new Color(-3355444));
                gbc = new GridBagConstraints();
                gbc.gridy = 1;
                gbc.weightx = 1.0;
                gbc.fill = GridBagConstraints.HORIZONTAL;
                gbc.insets = new Insets(0,0,0,0);
                ((GridBagLayout)jPanel1.getLayout()).setConstraints(labelChange, gbc);
                jPanel1.add(labelChange);
                labelClose = new com.sun.java.swing.JLabel();
                labelClose.setText("Close");
                labelClose.setBounds(0,25,113,25);
                labelClose.setFont(new Font("Dialog", Font.PLAIN, 11));
                labelClose.setForeground(new Color(0));
                labelClose.setBackground(new Color(-3355444));
                gbc = new GridBagConstraints();
                gbc.gridy = 2;
                gbc.weightx = 1.0;
                gbc.fill = GridBagConstraints.HORIZONTAL;
                gbc.insets = new Insets(0,0,0,0);
                ((GridBagLayout)jPanel1.getLayout()).setConstraints(labelClose, gbc);
                jPanel1.add(labelClose);
                labelLo = new com.sun.java.swing.JLabel();
                labelLo.setText("Lo");
                labelLo.setBounds(113,25,113,25);
                labelLo.setFont(new Font("Dialog", Font.PLAIN, 11));
                labelLo.setForeground(new Color(0));
```

```
labelLo.setBackground(new Color(-3355444));
gbc = new GridBagConstraints();
gbc.gridy = 2;
gbc.weightx = 1.0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.insets = new Insets(0,0,0,0);
((GridBagLayout)jPanel1.getLayout()).setConstraints(labelLo, gbc);
jPanel1.add(labelLo);
labelVolume = new com.sun.java.swing.JLabel();
labelVolume.setText("Vol");
labelVolume.setBounds(226,25,113,25);
labelVolume.setFont(new Font("Dialog", Font.PLAIN, 11));
labelVolume.setForeground(new Color(0));
labelVolume.setBackground(new Color(-3355444));
gbc = new GridBagConstraints();
gbc.gridy = 2;
gbc.weightx = 1.0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.insets = new Insets(0,0,0,0);
((GridBagLayout)jPanel1.getLayout()).setConstraints(labelVolume, gbc);
jPanel1.add(labelVolume);
jLabel20 = new com.sun.java.swing.JLabel();
jLabel20.setBounds(339,25,113,25);
jLabel20.setFont(new Font("Dialog", Font.PLAIN, 11));
jLabel20.setForeground(new Color(0));
jLabel20.setBackground(new Color(-3355444));
gbc = new GridBagConstraints();
gbc.gridy = 2;
gbc.weightx = 1.0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.insets = new Insets(0,0,0,0);
((GridBagLayout)jPanel1.getLayout()).setConstraints(jLabel20, gbc);
jPanel1.add(jLabel20);
labelBidPrice = new com.sun.java.swing.JLabel();
labelBidPrice.setText("Bid: 0.0");
labelBidPrice.setBounds(0,50,113,25);
labelBidPrice.setFont(new Font("Dialog", Font.PLAIN, 11));
labelBidPrice.setForeground(new Color(0));
labelBidPrice.setBackground(new Color(-3355444));
gbc = new GridBagConstraints();
gbc.gridy = 3;
gbc.weightx = 1.0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.insets = new Insets(0,0,0,0);
((GridBagLayout)jPanel1.getLayout()).setConstraints(labelBidPrice, gbc);
jPanel1.add(labelBidPrice);
```

```
        labelBidQty = new com.sun.java.swing.JLabel();
        labelBidQty.setText("Bid size: 0.0");
        labelBidQty.setBounds(113,50,113,25);
        labelBidQty.setFont(new Font("Dialog", Font.PLAIN, 11));
        labelBidQty.setForeground(new Color(0));
        labelBidQty.setBackground(new Color(-3355444));
        gbc = new GridBagConstraints();
        gbc.gridy = 3;
        gbc.weightx = 1.0;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(0,0,0,0);
        ((GridBagLayout)jPanel1.getLayout()).setConstraints(labelBidQty, gbc);
        jPanel1.add(labelBidQty);
        labelAskPrice = new com.sun.java.swing.JLabel();
        labelAskPrice.setText("Ask: 0.0");
        labelAskPrice.setBounds(226,50,113,25);
        labelAskPrice.setFont(new Font("Dialog", Font.PLAIN, 11));
        labelAskPrice.setForeground(new Color(0));
        labelAskPrice.setBackground(new Color(-3355444));
        gbc = new GridBagConstraints();
        gbc.gridy = 3;
        gbc.weightx = 1.0;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(0,0,0,0);
        ((GridBagLayout)jPanel1.getLayout()).setConstraints(labelAskPrice, gbc);
        jPanel1.add(labelAskPrice);
        labelAskQty = new com.sun.java.swing.JLabel();
        labelAskQty.setText("Ask size: 0.0");
        labelAskQty.setBounds(339,50,113,25);
        labelAskQty.setFont(new Font("Dialog", Font.PLAIN, 11));
        labelAskQty.setForeground(new Color(0));
        labelAskQty.setBackground(new Color(-3355444));
        gbc = new GridBagConstraints();
        gbc.gridy = 3;
        gbc.weightx = 1.0;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(0,0,0,0);
        ((GridBagLayout)jPanel1.getLayout()).setConstraints(labelAskQty, gbc);
        jPanel1.add(labelAskQty);
        //}}

        //MANUAL INIT_CONTROLS
        setMinimumSize(new Dimension(0,70));
        setPreferredSize(new Dimension(0,70));

paneTitleBar = new TitleBarPane( );
```

```
            paneTitleBar.setTitle( " Real-time quote shown below" );
            paneTitleBar.getButtonHelp( ).setVisible( false );
            paneTitleBar.getButtonClose( ).setVisible( false );
            paneTitleBar.setPreferredSize(new Dimension(0,18));
    paneTitleBar.setBorder(new MatteBorder(1,0,0,0,Color.black));
    add( BorderLayout.NORTH, paneTitleBar );
            //

            //{{REGISTER_LISTENERS
            SymAction lSymAction = new SymAction();
            paneTitleBar.addActionListener(lSymAction);
            //}}

            //MANUAL REGISTER_LISTENERS
            //
      }

  /**
   * Code to perform when this object is requested to initialize.
   */
      public void init( )
      {
    if ( registry == null )
    {
      // Complain about null registry.
      System.out.println( getClass( ).getName( ) + " --> Registry not set." );

      return;
    }

    // Create the default model.
    mktInfo = new MktInfo( registry );

    // Get reference to collection(s).
    witUsers = (WITUsers)registry.get( WITUsers.OT_WITUSER_COLLECTION );
    preferences = ( (Preferences)registry.get(
Preferences.OT_PREFERENCE_COLLECTION) );
    mktInfos = ( (MktInfos)registry.get( MktInfos.OT_MKTINFO_COLLECTION) );
    issues = ( (Issues)registry.get( Issues.OT_ISSUE_COLLECTION) );

    // Register with observables.
    mktInfos.addObserver( this );
  }

  /**
   * Code to perform when this object is requested to shutdown.
```

```
*/
    public void shutdown( )
    {
}

/**
 * Code to perform when this object is garbage collected.
 */
    public void finalize( ) throws Throwable
    {
  // De-register with observables.
  mktInfos.deleteObserver( this );

  // If you have a reference to it, free it.
  witUsers = null;
  preferences = null;
  mktInfos = null;
  issues = null;
  mktInfo = null;

  // If applications is in debug mode.
  if ( registry.getIsDebug( ) == true )
  {
    System.out.println( this.getClass( ).getName( ) + " in method finalize - I have
been garbage collected." );
  }

  registry = null;

      super.finalize( );
    }

/**
 * Serialize the user's properties.
 */
    public void serialize( String setName )
    {
  WITUser user = (WITUser)witUsers.getLoggedOnUser( );
  if ( user == null )
  {
    if ( registry.getIsDebug( ) == true )
    {
      System.out.println( getClass( ).getName( ) +
        " ---> No logged on user for preference serialization." );
    }
```

```
        return;
      }

      Vector v = new Vector( 1 );

      Preference preference;
      preference = new Preference( registry );
      preference.setUserEntityId( user.getEntityId( ) );
      preference.setSetName( setName );
      preference.setPreferenceName( getClass( ).getName( ) + "_isShowing" );
      preference.setValue( String.valueOf( isShowing( ) ) );
      v.addElement( preference );

      TagMessage reqAddMemberMsg = null;
      Preference newPreference;
      Preference oldPreference;
      Enumeration elements = v.elements( );
      while( elements.hasMoreElements( ) )
      {
        preference = (Preference)elements.nextElement( );

        oldPreference =
(Preference)preferences.findUsingUserEntityIdSetNamePreferenceName(
        user.getEntityId( ), setName, preference.getPreferenceName( ) );

        if ( oldPreference == null )
        {
          try
          {
            reqAddMemberMsg = preferences.addMember( preference, null );
          }
          catch ( AddMemberException e )
          {
            if ( registry.getIsDebug( ) == true )
              System.out.println( e );
          }
          catch ( NoConnectionException e )
          {
            if ( registry.getIsDebug( ) == true )
              System.out.println( e );
          }
          catch ( NullPointerException e )
          {
            if ( registry.getIsDebug( ) == true )
              System.out.println( e );
          }
```

```
        }
        else
        {
          if ( !preference.queryIsChanged( oldPreference ) )
          {
            continue;
          }

          newPreference = new Preference( registry );
          newPreference.duplicateAllFrom( oldPreference );
          newPreference.setValue( preference.getValue( ) );

          try
          {
            reqAddMemberMsg = preferences.updateMember( newPreference, null );
          }
          catch ( UpdateMemberException e )
          {
            if ( registry.getIsDebug( ) == true )
              System.out.println( e );
          }
          catch ( NoConnectionException e )
          {
            if ( registry.getIsDebug( ) == true )
              System.out.println( e );
          }
          catch ( NullPointerException e )
          {
            if ( registry.getIsDebug( ) == true )
              System.out.println( e );
          }
        }

        setCursor( Cursor.getPredefinedCursor( Cursor.WAIT_CURSOR ) );
        while ( true )
        {
          if ( reqAddMemberMsg.getResponseResultCode( ) !=
TagMessage.MSG_RESULT_None )
          {
            break;
          }

          try
          {
            Thread.sleep( 100 );
          }
```

```
            catch ( InterruptedException e )
            {
              break;
            }
          }
          setCursor( Cursor.getPredefinedCursor( Cursor. DEFAULT_CURSOR ) );

          int msgResultCode = reqAddMemberMsg.getResponseResultCode( );
          if ( msgResultCode != TagMessage.MSG_RESULT_NoError )
          {
            JOptionPane.showMessageDialog( JOptionPane.getFrameForComponent( this
),
                "Failed to add preference in " + getClass( ).getName( )
                + preference.getPreferenceName( ) + ".\n\n"
                +"result = " + msgResultCode, "Notice",
JOptionPane.INFORMATION_MESSAGE );

            break;
          }
        }
      }

      /**
       * Deserialize the user's properties.
       */
          public void deserialize( String nameSet )
          {
        Preference preference;

        WITUser user = (WITUser)witUsers.getLoggedOnUser( );
        if ( user == null )
        {
          if ( registry.getIsDebug( ) == true )
          {
            System.out.println( getClass( ).getName( ) +
              " ---> No logged on user for preference serialization." );
          }

          return;
        }

        try
        {
          preference = preferences.findUsingUserEntityIdSetNamePreferenceName(
  user.getEntityId( ),
              nameSet, getClass( ).getName( ) + "_isShowing" );
```

```
        boolean isShowing = Boolean.valueOf( preference.getValue( ) ).booleanValue( );
        setVisible( isShowing );
      }
      catch ( NullPointerException e )
      {
      }
        }

  /**
   * Called when observers in the observable list need to be updated.
   * @ param o - the list of observers.
   * @ param arg - the argument being notified.
   */
  public void update( Observable o, Object arg )
  {
     // Get the change event.
     ObservableChange observableChange = (ObservableChange)arg;

     if ( observableChange.isAdd( ) )
     {
     }
     else
     if ( observableChange.isChange( ) )
     {
        // Get the MktInfo which changed.
        MktInfo mktInfo = (MktInfo)observableChange.getMember( );

        if ( mktInfo == null )
        {
           return;
        }

        // Test if the issue which has change is the one showing
        // in this view.
        if ( mktInfo.getEntityId( ) == this.mktInfo.getEntityId( ) )
        {
           // The underlying model for this view has changed.
           // Set the model for this view to reflect the changes
           // of the underlying model.
           setModel( mktInfo );
        }
     }
      if ( observableChange.isDelete( ) )
      {
        // Set the default model.
        setModel( new MktInfo( registry ) );
```

```
    }
    else
    if ( observableChange.isSelected( ) )
    {
    }
    else
    if ( observableChange.isTick( ) )
    {
    }
  }

  /**
   * Sets the MktInfo model displayed in this view.
   * @ param mktInfo - the model displayed in this view.
   */
  public void setModel( MktInfo mktInfo )
  {
    // Duplicate the real issue.
    this.mktInfo.duplicateAllFrom( mktInfo );

    // The question as to when to paint this component. Should the caller changing
    // the model call paint or should the view assume a paint when a new model is set?
    // For now the decision is when the model changes assume a paint if preferred.
    paint( );
  }

  /**
   * Gets the MktInfo model displayed in this view.
   * @ return MktInfo - the model displayed in this view.
   */
  public MktInfo getModel( )
  {
    return mktInfo;
  }

  /**
   * Diaplays the MktInfo's data on the screen.
   * @ param mktInfo - the new model to display in this view.
   */
  public void paint( MktInfo mktInfo )
  {
    // Set the default model.
    setModel( mktInfo );
  }

  /**
```

```
    * Diaplays the issue's market data on the screen.
    */
   public void paint( )
   {
      Issue issue = null;

      // Get Issue for the MktInfo model.
      issue = (Issue)issues.findUsingEntityId( mktInfo.getIssueEntityId( ) );

      if ( issue == null )
      {
         issue = new Issue( registry );
      }

      // Get the market information from model.
      String openPrice = "Open  " + Util.toString( "zzzn.nntttt", (short)6,
         mktInfo.getOpenPrice( ) );
      String closePrice = "Close " + Util.toString( "zzzn.nntttt", (short)6,
         mktInfo.getClosePrice( ) );
      String hi = "Hi " + Util.toString( "zzzn.nntttt", (short)6,
         mktInfo.getHiPrice( ) );
      String lo = "Lo " + Util.toString( "zzzn.nntttt", (short)6,
         mktInfo.getLoPrice( ) );
      String bidPrice = "Bid: " + Util.toString( "zzzn.nntttt", (short)6,
         mktInfo.getBidPrice( ) );
      String bidQty = "Bid size: " + Util.toString( "zzz,zzn", (short)0,
         mktInfo.getBidQty( ) );
      String askPrice = "Ask: " + Util.toString( "zzzn.nntttt", (short)6,
         mktInfo.getAskPrice( ) );
      String askQty = "Ask size: " + Util.toString( "zzz,zzn", (short)0,
         mktInfo.getAskQty( ) );
      String lastPrice = "Last " + Util.toString( "zzzn.nntttt", (short)6,
         mktInfo.getLastPrice( ) );
      String change = "Change " + Util.toString( "+zzzz.nntttt", (short)6,
         mktInfo.getCloseToLastChange( ) );
      String volume = "Vol " + Util.toString( "zzz,zzz,zzz", (short)0,
         0 ); //mktInfo.getVolume( )

      // Show the market information.
      labelSymbol.setText( issue.getSymbol( ) );
      labelDescription.setText( issue.getDescription( ) );
      labelOpen.setText( openPrice );
      labelClose.setText( closePrice );
      labelHi.setText( hi );
      labelLo.setText( lo );
      labelBidPrice.setText( bidPrice );
```

```
    labelBidQty.setText( bidQty );
    labelAskPrice.setText( askPrice );
    labelAskQty.setText( askQty );
    labelLastPrice.setText( lastPrice );
    labelChange.setText( change );
    labelVolume.setText( volume );
  }

  /**
   * Sets the registry for this object.
   * @ param Registry - The collection of registered objects.
   */
  public void setRegistry( Registry registry )
  {
    this.registry = registry;
  }

  /**
   * Shows or hides this component depending on the value of parameter b.
   * @ param b - If true, shows this component; otherwise, hides this component.
   */
  public void setVisible( boolean aFlag )
  {
    super.setVisible( aFlag );

    // Add this component as a dirty region. Forces parent to layout.
    invalidate( );
  }

  public TitleBarPane getTitleBarComponent( )
  {
    return paneTitleBar;
  }

  public JPanel getIssueInfoComponent( )
  {
    return paneIssueInfo;
  }

  /**
   * Sets the childrens foreground to color.
   * @ param color - the new color of the children's foreground
   */
      public void setForeground( Color c )
      {
    super.setForeground( c );
```

```
    if ( paneTitleBar != null ) paneTitleBar.getLabelTitle( ).setForeground( c );
    if ( labelSymbol != null ) labelSymbol.setForeground( c );
    if ( labelDescription != null ) labelDescription.setForeground( c );
    if ( labelOpen != null ) labelOpen.setForeground( c );
    if ( labelOpen != null ) labelOpen.setForeground( c );
    if ( labelHi != null ) labelHi.setForeground( c );
    if ( labelLastPrice != null ) labelLastPrice.setForeground( c );
    if ( labelChange != null ) labelChange.setForeground( c );
    if ( labelClose != null ) labelClose.setForeground( c );
    if ( labelLo != null ) labelLo.setForeground( c );
    if ( labelVolume != null ) labelVolume.setForeground( c );
    if ( jLabel20 != null ) jLabel20.setForeground( c );
    if ( labelBidPrice != null ) labelBidPrice.setForeground( c );
    if ( labelBidQty != null ) labelBidQty.setForeground( c );
    if ( labelAskPrice != null ) labelAskPrice.setForeground( c );
    if ( labelAskQty != null ) labelAskQty.setForeground( c );
      }

public void addHelpListener( HelpListener l )
{
  listenerList.add( HelpListener.class, l );
}

public void removeHelpListener( HelpListener l )
{
  listenerList.remove( HelpListener.class, l );
}

// Notify all listeners that have registered interest for
// notification on this event type.  The event instance
// is lazily created using the parameters passed into
// the fire method.
protected void firerequestHelp( HelpEvent helpEvent )
{
  // Guaranteed to return a non-null array
  Object[] listeners = listenerList.getListenerList( );
  // Process the listeners last to first, notifying
  // those that are interested in this event
  for ( int i = listeners.length-2; i>=0; i-=2 )
  {
    if ( listeners[i] == HelpListener.class )
    {
      ((HelpListener)listeners[i+1]).requestHelp( helpEvent );
    }
  }
```

```
}

    //{{DECLARE_CONTROLS
    com.sun.java.swing.JPanel jPanel1;
    com.sun.java.swing.JPanel paneIssueInfo;
    com.sun.java.swing.JLabel labelSymbol;
    com.sun.java.swing.JLabel labelDescription;
    com.sun.java.swing.JLabel jLabel2;
    com.sun.java.swing.JLabel jLabel3;
    com.sun.java.swing.JLabel labelOpen;
    com.sun.java.swing.JLabel labelHi;
    com.sun.java.swing.JLabel labelLastPrice;
    com.sun.java.swing.JLabel labelChange;
    com.sun.java.swing.JLabel labelClose;
    com.sun.java.swing.JLabel labelLo;
    com.sun.java.swing.JLabel labelVolume;
    com.sun.java.swing.JLabel jLabel20;
    com.sun.java.swing.JLabel labelBidPrice;
    com.sun.java.swing.JLabel labelBidQty;
    com.sun.java.swing.JLabel labelAskPrice;
    com.sun.java.swing.JLabel labelAskQty;
    //}}

    class SymAction implements java.awt.event.ActionListener
    {
        public void actionPerformed(java.awt.event.ActionEvent event)
        {
            Object object = event.getSource();
            if (object == paneTitleBar.getButtonHelp( ))
                paneTitleBarButtonHelp_actionPerformed(event);
        }
    }

    void paneTitleBarButtonHelp_actionPerformed(java.awt.event.ActionEvent
event)
    {
      firerequestHelp( new HelpEvent( this, 0 ) );
    }
}
```